OrangeAssassin Documentation

Release 1.0a1

SpamExperts

Mar 28, 2017

Contents

Contents:

CHAPTER 1

Introduction

OrangeAssassin is an open-source drop-in replacement for SpamAssassin.

Compatibility

OrangeAssassin is compatible with the following Python versions:

- Python 2.7
- Python 3.2 and later
- PyPy
- PyPy3

Contribute

- Issue Tracker
- Source Code

Getting the source

To clone the repository using git simply run:

```
git clone https://github.com/SpamExperts/OrangeAssassin
```

Please feel free to fork us and submit your pull requests.

Running tests

To run the project's tests you will need to first:

1. Create a python virtualenv and activate it *(Recommended only)*
2. Clone the repository from GitHub.
3. Install sqlalchemy or pymysql package.

4. Install the base dependencies from *requirements/base.txt* with pip
5. Install the the dependencies for the python version you are using from the *requirements* folder
6. Install the dependencies for running tests from *requirements/tests.txt*
7. Download the GeoIP databases (for IPv4 and IPv6)
8. Run the *setup.py* script

Note: Some requirements (e.g. Pillow) require some additional build dependencies when installing them.

The OrangeAssassin tests are split into *unittest* and *functional* tests.

Unitests perform checks against the current source code and **not** the installed version of OrangeAssassin. To run all the unittests suite:

```
py.test tests/unit/
```

Functional tests perform checks against the installed version of OrangeAssassin and **not** the current source code. These are more extensive and generally take longer to run. They also might need special setup. To run the full suite of functional tests:

```
env USE_PICKLES=0 py.test tests/functional/ (or py.test tests/functional/)
```

If you want to compile rules and avoid re-parsing:

    env USE_PICKLES=1 py.test tests/functional/

Or you can run *all* the tests with just:

```
py.test
```

An example for Python3 would be:

```
sudo apt-get install python3-dev libjpeg-dev build-essential zlib1g-dev
virtualenv -p /usr/bin/python3 ~/oa-env
source ~/oa-env/bin/activate
git clone https://github.com/SpamExperts/OrangeAssassin
cd OrangeAssassin
pip install sqlalchemy || pip install pymysql
pip install -r requirements/base.txt
pip install -r requirements/python3.txt
pip install -r requirements/tests.txt
wget http://geolite.maxmind.com/download/geoip/database/GeoLiteCountry/GeoIP.dat.gz
gunzip GeoIP.dat.gz
wget http://geolite.maxmind.com/download/geoip/database/GeoIPv6.dat.gz
gunzip GeoIPv6.dat.gz
python setup.py install
env USE_PICKLES=0 py.test
env USE_PICKLES=0 py.test
```

Note: See also the *.travis.yml* file where all these instructions are set for the automatic builds.

Building documentation

In order to build the documentation based on the docs files from the repository:

1. Run the same steps for running the tests (including installing all requirements, installing OrangeAssassin, etc.).
2. Install the documentation libraries from *requirements/docs.txt*
3. Change directory to *docs*
4. Run *make html*
5. The HTML version of the documentation will be generated in the *docs/_build/* directory.

See also the helper script *docs/generate_plugin_doc.py* that generates a documentation page for the specified plugin. After adding a new plugin:

- Use the script to generate a new page for it
- Add a reference to the list from *docs/plugins.rst*
- Add autodoc to *docs/pad.plugins.rst*

License

CHAPTER 2

Configuration

This page describes how to configure OrangeAssassin.

Configuration files

The OrangeAssassin configuration can be separated into multiple files. These are read from the *configpath* and *sitepath* directories. You can change these locations using the *-C* or *-S* options of the daemon and CLI script.

More files can be included from other locations by using the include option:

```
# This includes a different file
include /etc/orangeassassin/custom_prefs.cf
```

Users can also configure custom preferences in their home directory when running the CLI script. This location is also customizable with the *-P* option. Note that daemon does NOT accept user preferences by default and you will have to enable it with *allow_user_rules*.

Note: The order of the files IS important as it determines the order of the rules loading and executing. To change the order in which the rules are checked see the *priority rule option*.

Configuration types

OrangeAssassin accepts various types of configuration options. The current types are:

int Integer number.

float Floating point number.

bool Boolean value, can be one of: 1, 0, True, False.

str A simple string value.

list A comma separated list of strings. For example defining this:

```
pyzor_servers public.pyzor.org,my.pyzor.example.com
```

Will be evaluated as:

```
["public.pyzor.org", "my.pyzor.example.com"]
```

Defining the same option multiple time WILL override the previous setting.

append This option can be specified multiple times without overriding previous settings. Every time the option is specified the values are appended to a list. For example for the report option:

```
report This message was marked as spam on _HOSTNAME_.
report The message score was _SCORE_.
report Contact me at _CONTACTADDRESS_.
```

Will result in the final option being evaluated as:

```
["This message was marked as spam on _HOSTNAME_.",
 "The message score was _SCORE_.",
 "Contact me at _CONTACTADDRESS_."]
```

clear Clears one or more of the append type option.

Compiling rules

Users can compile rules in OrangeAssassin in two ways:

1. Re-parsing the rules:

```
$ ./scripts/match.py -t -C /root/myconf/ --sitepath /root/myconf/ < /root/test.eml
```

2. Avoiding re-parsing, in order to use this ability, users should:

   2.1. Run compile.py with -sp flag to specify the path for the file where the serialization will be done:

```
$ ./scripts/compile.py -t -C /root/myconf/ --sitepath /root/myconf/ < /root/test.eml -se
```

   2.2. Run match.py using:

   - se (use serialization)
   - sp (specify the path for the file where the serialization was done)

Note: Users can use the ability to compile rules and avoid re-parsing only if they have:

- Python 2.7.11 or later
- Python 3.5 or later

Options

Filtering options

required_score 5.0 (type *float*) Set a minimum required score for a message for it to be treated as spam.

use_bayes True (type *bool*) Controls whether or not the bayesian filter should be checked.

use_network True (type *bool*) Controls whether or not network checks should be perfomed on the message.

envelope_sender_header ["X-Sender", "X-Envelope-From", "Envelope-Sender", "Return-Path", "From"] (type *append*) Specifies which header should be used when determining the envelope sender of the message.

allow_user_rules False (type *bool*) If set to True the daemon will also load user preferences. Note that this can be a possible security risk, which is why it's disabled by default.

Message modifications

add_header [] (type *append*) Adds one header to the message. The value for this option must be in the following format:

```
add_header [all|spam|ham] [header_name] [header_value]
```

If the first argument is *all* then the header is added to ALL messages. Otherwise the header is added only to messages that were classified as spam or ham. Note that the header name will be append with *X-Spam-* and the header string ill have any TAGS replaced with their values. For example:

```
add_header all OA-Report Checked with OrangeAssassin _SCORE_
```

Will add a new header to every message like:

```
X-Spam-OA-Report: Checked with OrangeAssassin <score>
```

remove_header [] (type *append*) Removes all header from message with the specified name. The value for this option must be in the following format:

```
remove_header [all|spam|ham] [header_name]
```

clear_headers N/A (type *clear*) Clear all previously set options that add or remove headers (i.e. any from *add_header* or *remove_header*).

Reporting

report [] (type *append*) A list of strings that form the report. The report can be returned when the CLI script is called with *-t* and is also included by default in messages that have been marked as spam. Note that this string will have any TAGS replaced with their values.

clear_report_template N/A (type *clear*) Clear the report list.

report_safe 1 (type *int*) When this option is set to 0 only header modification are made to the messages. In addition an X-Spam-Report will be added to the messages that contains the *report* for this message. Note this only applies to messages classified as spam.

When this option is set to 1 and the messages is marked as spam, OrangeAssassin will generate a multipart/mixed messages. The new message will have *text/plain* part with the OrangeAssassin report and *message/rfc882* part with the original message.

When the option is set to 2 instead of using a *messages/rfc882* content type, a text/plain one will be used instead.

report_contact None (type *str*) Set the contact address that is exposed in the *_CONTACTADDRESS_* tag.

Network Options

Syntax:

```
trusted_networks [!]IP_ADDRESS[/MASKLEN] [...]

internal_networks [!]IP_ADDRESS[/MASKLEN] [...]

msa_networks [!]IP_ADDRESS[/MASKLEN] [...]
```

! excludes the network from the list

MASKLEN the CIDR-style netmask length specified in bits. If it's not specified it will be deduced from the IP_ADDRESS

IP_ADDRESS an IPv4 or IPv6 address optionally enclosed in square brackets. If no masklen is specified then one will be deduced from the ip like this: If the ip has less than 4 octets and ends with a trailing dot then the masklen is *num_octets * 8* if there is no trailing dot then the mask will be *32* for IPv4 addresses and *128* for IPv6 addresses

trusted_networks [] (type *append split*) You can specify multiple networks. With each network specified, it will be added to the list of trusted networks.

The networks are searched sequentially with the first match stopping the search, so you should write more specific subnets first.

Note: 127.0.0.0/8 and ::1 are always included in trusted_networks and cannot be overriden

---

Trusted networks in our case means that a relay host from one of these networks is considered out of the control of spammers, open relays, or open proxies. A trusted network could relay spam but spam will not originate from it and it will not forge header data. So we will not do dns blacklist checks for any host in these networks

This setting should define the networks that you trust but are not internal relays or MXes for your domains

Examples:

```
# Trust all in 192.168.*.*

trusted_networks 192.168.

# or

trusted_networks 192.168.0.0/16

# Trust all in 192.168.*.* except those in 192.168.1.*

trusted_networks !192.168.1. 192.168.

# or

trusted_networks !192.168.1.0/24 192.168.0.0/16

# or

trusted_networks !192.168.1.0/24
trusted_networks !192.168.0.0/16
```

clear_trusted_networks N/A (type *clear*) Empties the list of trusted networks. 127.0.0.0/8 and ::1 will still exist and they cannot be removed

internal_networks [] (type *append split*) When you define an internal network then all hosts in the network are considered to be MXes for your domains or internal relays.

Internal networks are a subset of trusted networks so they will be added as a trusted network too

If trusted networks is set and internal_networks is not then trusted networks will also be considered internal networks.

Note: 127.0.0.0/8 and ::1 are always included in trusted_networks and cannot be overriden

clear_internal_networks N/A (type *clear*) Empties the list of internal networks. 127.0.0.0/8 and ::1 will still exist and they cannot be removed

msa_networks [] (type *append split*) MSA hosts, also known as MX relays are hosts that accept mail from your own users and authenticate them properly.

These relays will never accept mail from hosts that aren't authenticated in some way. If an MSA relays is found then all relays after it will get the same internal/trusted classification as that one

When using msa_networks to identify an MSA host it is recommended to treat it as both trusted and internal. When an MSA is also acting as an MX or an intermediate relay you must always treat it as both trusted and internal and make sure that the MSA includes visible auth tokens in it's Received header

Warning: You shouldn't include an msa that is also an MX or an intermediate relay for an MX in this setting because it will result in uknown external relays being trusted

clear_msa_networks N/A (type *clear*) Empties the list of msa networks

DNS

dns_server [] (type *append*) Specify a list of nameservers to query when doing DNS lookups. These can specified as IPv4 or IPv6 address with an optional port followed. Example:

```
dns_server 127.0.0.1

dns_server 127.0.0.1:53

dns_server [::1]:53
```

If no such nameserver is specified, the default ones from */etc/resolv.conf* will be used.

clear_dns_servers N/A (type *clear*) Clear any custom nameserver set by *dns_server*.

default_dns_lifetime 10.0 (type *float*) Sets the timeout for a full DNS lookup. I.e. any DNS lookup will have at most 10 seconds to get a valid response from one of the DNS server.

default_dns_timeout 2.0 (type *float*) Set the timeout for a DNS lookup from a single nameserver.

dns_available yes (type *str*) Configure whether DNS resolving is available or not. If you specify it as yes or no then no tests will be performed. Example:

```
dns_available yes

dns_available no
```

If you want to determine the availability dynamically you can use the value *test* or *test: domain1 domain2 ... domainN*. In that case a query will be performed for three of the domain names given chosen at random. If any of them gives a response then dns will be considered available. The test will be performed again according to the :ref: *dns_test_interval option <dns_test_interval>* Example:

```
dns_available test:domain1 domain2 domain3 domain4
```

If no domains are specified with the test option then a default list will be used Example:

```
dns_available test
```

dns_test_interval 600s (type *str*) If you set the *dns_available option* to *test* then by setting, the actual test will be performed no sooner that the interval you set here. You can set just a number or a number with a suffix to determine the the time unit (s, m, h, d, w) Example:

```
dns_test_interval 600

dns_test_interval 600s

dns_test_interval 10m
```

dns_query_restriction "" (type *string*) Configure restrictions for querying the dns. Almost all dns queries are subject to the dns_query_restriction. Before performing a query the domain is tested against these restrictions and when a match occurs the query is performed according to the allow/deny setting for that match. If no match is found then the query is allowed by default.

When testing a domain it's labels are stripped successively to check if a parent matches.

All of the following would be denied example.com, 1.example.com 1.2.example.com

```
dns_query_restriction deny example.com
```

This way 1.example.com 2.1.example.com would be denied but example.com would be allowed

```
dns_query_restriction deny 1.example.com
```

You can deny a wider group of domains and only allow one subgroup like this:

```
dns_query_restriction deny example.com
dns_query_restriction allow 1.example.com
```

In this case example.com and all of its subdomains would be denied except 1.example.com and all of it's subdomains which would be allowed

Tags

Template tags

The following tags can be used as placeholders in certain options. They will be replaced by the corresponding value when they are used.

YESNOCAPS "YES"/"NO" for is/isn't spam

YESNO "Yes"/"No" for is/isn't spam

REQD Message threshold

VERSION version (eg. 1.0a)

SUBVERSION sub-version/code revision date (eg. 2016-01-15)

HOSTNAME Hostname of the machine the mail was processed on

TESTS(,) tests hit separated by "," (or other separator)

PREVIEW content preview

REPORT terse report of tests hit (for header reports)

SUMMARY summary of tests hit for standard report (for body reports)

CONTACTADDRESS Contents of the 'report_contact' setting

Received Headers tags

These are metadata parsed from the last received header (top most) and exposed in tags which can be accessed with the next keywords:

RDNS Reverse DNS made automatically by MTA

HELO Helo identification

_IP_ Relay IP address

ENVFROM For routing the received e-mail to the intended recipient(s)

_BY_ Mail server name and system: domain of the server receiving the e-mail

IDENT Ident lookup

_ID_ Message identification number given by the machine who received the message

AUTH Authentication

RELAYSTRUSTED Relays used and deemed to be trusted

RELAYSUNTRUSTED Relays used that can not be trusted

RELAYSINTERNAL Relays used and deemed to be internal

RELAYSEXTERNAL Relays used and deemed to be external

LASTEXTERNALIP IP address of client in the external-to-internal SMTP handover

LASTEXTERNALRDNS Reverse-DNS of client in the external-to-internal SMTP handover

LASTEXTERNALHELO HELO string used by client in the external-to-internal SMTP handover

CHAPTER 3

Daemon

This page explains how to run OrangeAssassin in daemon mode.

Starting the daemon

Starting the daemon can be done by running the `oad.py` script with the `--daemonize` option and specifying a pidfile:

```
oad.py -d -r /var/run/oad.pid
```

It's also recommended to active preforking with an appropriate number of workers depending on your system:

```
oad.py -d -r /var/run/oad.pid --prefork 4
```

Depending on your distribution you might also want to change the path to the configuration directory and the site configuration directory. E.g:

```
oad.py -d -r /var/run/oad.pid --prefork 4 -C /usr/share/spamassassin -S /etc/mail/spamassassin
```

You can also change the port and IP on which the daemon is listenting on:

```
oad.py -d -r /var/run/oad.pid --prefork 4 -i 127.0.0.2 -p 30783
```

For more info see the `--help` option of the script.

Reloading the daemon

Reloading the daemon can be achieved by sending the USR1 signal to the main process OR by using the option of the `oad.py` script:

```
oad.py -r /var/run/oad.pid reload
```

Stopping the daemon

Gracefully stopping the daemon and the workers can be achieved by sending the TERM signal to the main process OR by using the option of the `oad.py` script:

```
oad.py -r /var/run/oad.pid stop
```

CHAPTER 4

Writing Rules

The rules configuration defines the checks that are done on the messages. Each rule has a unique identifier, writen in all caps, and can have multiple options.

After all rules are checked a final score is provided for the message and according to the *required_score* option the message is marked as spam.

Defining a rule

Every rule must be in the following format:

```
<rule type>     <rule identifier>   <value>
```

Simple rule definition example:

```
body        LOOK_FOR_TEST   /test/
```

Where *body* is the rule type, *LOOK_FOR_TEST* is the identifier and the value is */test/*. This rule will look for the string "test" in the body of the message and the rule will be triggered if it is found. When a rule is triggered the corresponding score is added to the total score of the message.

For every message all defined rules are checked and the score applied with the following exceptions:

- Any rule that has an identifier starting with __ will not be checked.
- Any rule that has a score of *0* will not be checked.

Note: Rules that are not checked can still be used in combination with other rules. See the *meta* rule type for more details.

Rule options

Additional options can be configured to any rule in the following format:

```
<option name> <rule identifier> <value>
```

The parser will use the unique identifier to configure the option to the specific rule with the same name. The option **doesn't** have to be added immediately after the rule definition (i.e. the next line), but it has to be somewhere after the initial rule definition.

Note: Defining the same rule or option twice **will override** the previous value.

Scoring option

Any rule defined will have by default a score of **1.0**. This can be adjusted by using the score option:

- A positive score means that the message is more likely to be spam
- A negative score means that the message is more likely to be legitimate
- A score of *0* disables the rule

Examples:

```
body    LOOK_FOR_TEST /test/
score   LOOK_FOR_TEST 1.5

header  LOOK_FOR_SUBJECT_TEST Subject =~ /test/
score   LOOK_FOR_SUBJECT_TEST -5
```

More advance scoring can be specified for any rule depending on whether the Bayesian classifier and network tests are activated. For example:

```
body    LOOK_FOR_TEST /test/
score   LOOK_FOR_TEST 1 1.5 0.5 3
```

For the advanced scoring the following final score will be used:

- The first score if the Bayesian classifier and networks tests are disabled (for this case *1*)
- The second score if the Bayesian classifier is disabled but the networks tests are enabled (for this case *1.5*)
- The third score if the Bayesian classifier is enabled but the networks tests are disabled (for this case *0.5*)
- The fourth score if the Bayesian classifier and the networks tests are enabled (for this case *3*)

Note: This configuration is optional and any rule that doesn't have it will get the default score of *1.0*.

Describe option

The describe option can be used to provide a small text that describes what the rule is doing. This text is useful when debugging and when generating various reports.

Example configuration:

```
report ==== Start report ====
report _REPORT_

body        LOOK_FOR_TEST /test/
describe    LOOK_FOR_TEST Look for the test string in the body.
```

And the result for a message that matches:

```
$ ./scripts/match.py -t -C /root/myconf/ --sitepath /root/myconf/ < /root/test.eml
Subject: Do you think this is Spam?

This is a test.


==== Start report ====

* 1.0 LOOK_FOR_TEST BODY: Look for the test string in the body.
```

For more details on the report see the report section of the documentation.

Note: This configuration is optional and any rule that doesn't have it will get "No description available".

Priority option

This option can be used to prioritize rules to be evaluated before others. By default the rules are checked in the order they are defined in the config file and their priority value is 0. A negative priority will leave the evaluation at the end. Also note that the value of the priority must be integer.

Example configuration:

```
body     TEST_RULE1  /test/
body     TEST_RULE2  /test/
body     TEST_RULE3  /test/
priority TEST_RULE2 5
priority TEST_RULE1 -1
```

They will be evaluated in the next order:

```
TEST_RULE2
TEST_RULE3
TEST_RULE1
```

Note: This configuration is optional and any rule that doesn't have it will get the priority 0.

Lang option (Locali[sz]ation)

The lang option can be used to provide text in a specific language. A line starting with the text lang xx will only be interpreted if the user is in that locale, allowing test descriptions and templates to be set for that language.

Rule option that enables using localized translations for rule descriptions and reports:

The locales string should specify either both the language and country, e.g. lang pt_BR, or just the language, e.g. lang de.

lang nl describe <RULE IDENTIFIER> <translated text> lang nl report <translated text>

Example configuration:

```
report ==== Start report ====
report _REPORT_

body        LOOK_FOR_TEST /test/
describe    LOOK_FOR_TEST Look for the test string in the body.
lang en describe    LOOK_FOR_TEST Description in en.
lang en report      Look for the test string in the body.
```

And the result for a message that matches:

```
$ ./scripts/match.py -t -C /root/myconf/ --sitepath /root/myconf/ < /root/test.eml
Subject: Do you think this is Spam?

This is a test.


==== Start report ====

Look for the test string in the body.
* 1.0 LOOK_FOR_TEST BODY: Description in en.
```

For more details on the report see the report section of the documentation.

Note: lang nl describe <RULE IDENTIFIER> <translated text> If the language specified as a second parameter correspond with locales, description for RULE IDENTIFIER will be overwritten.

Tflags option

Used to set flags on a test. These flags are used in the score-determination back end system for details of the test's behaviour.

tflags <TEST_NAME> <net|nice|learn|userconf|noautolearn>

net The test is a network test, and will not be run in the mass checking system or if -L is used, therefore its score should not be modified.

nice The test is intended to compensate for common false positives, and should be assigned a negative score.

userconf The test requires user configuration before it can be used.

learn The test requires training before it can be used.

noautolearn The test will explicitly be ignored when calculating the score for learning systems.

Example configuration:

```
report ==== Start report ====
report _REPORT_

body        LOOK_FOR_TEST /test/
tflags      LOOK_FOR_TEST nice
```

And the result for a message that matches:

```
$ ./scripts/match.py -t -C /root/myconf/ --sitepath /root/myconf/ < /root/test.eml
Subject: Do you think this is Spam?
```

```
This is a test.


==== Start report ====

* -1.0 LOOK_FOR_TEST BODY
```

Note: This configuration is optional and any rule that doesn't have it will get the default value False.

CHAPTER 5

Rule Types

Full Rule

Example rule:

```
full        NULL_IN_MESSAGES        /\x00/
describe    NULL_IN_MESSAGES        Message has NULL characters.
score       NULL_IN_MESSAGES        0.5
```

The *full* rule type matches a regular expression against the full raw message. This means that no parts are decoded and the message is in the same format as it was received.

Body Rules

The body rules will perform checks on the body part of the message. This means that anything after the headers is included in the check.

Body Type

Example rule:

```
body        LOOK_FOR_SPAM       /spam/
describe    LOOK_FOR_SPAM       Message has spam in it's text.
```

The *body* rule type matches a regular expression against extracted text of the message. The message is decoded and only the text parts are included when matching.

The message is:

- decoded and stripped of any headers
- all line break replaced with a single space
- all HTML tags removed
- subject headers prepended

RawBody Type

Example rule:

```
rawbody     LOOK_FOR_SPAM       /spam/
describe    LOOK_FOR_SPAM       Message has spam in it's raw text.
```

A similar variant of this check is *rawbody*, which unlike *body* matches the regular expression against the raw body of the message, without decoding any parts or removing any HTML tags.

The message is:

- decoded and stripped of any headers

Header Rule

The header rule will match regular various regular expression only against one or more headers of the message. The body of the message is completely ignored.

The are generally defined in the following format:

```
header <rule identifier> <header name> <match operator> <regex>
```

Where the <*match operator*> can be either:

- =~ for positive matching, i.e. the rule matches if the regex matches
- *!*~ for negated matching, i.e. the rule matches if the regex doesn't match

---

Note: If a message has multiple headers with the same name, **all** headers are verified.

Header Type

Example Rule:

```
header      LOOK_FOR_SUBJECT_SPAM   Subject =~ /spam/
describe    LOOK_FOR_SUBJECT_SPAM   Message has "spam" in Subject.
```

Note that the check is done on the decoded version of the subject and not on the raw version. For example for a header like:

```
Subject: =?utf8?B?VGhpcyBzcGFtIGlz?=
```

The check will be done on:

```
Subject: This spam is
```

Modifiers

For the header rules you can also append various modifiers to the header name. These will change the string on which the check is done.

- The **RAW** modifier will perform the check on the raw header instead of the decoded version. Example:

```
    header      UTF8_ENCODED_SUBJECT   Subject:raw =~ /^=?utf8?/
    describe    UTF8_ENCODED_SUBJECT   Subject is encoded with UTF-8
    score       UTF8_ENCODED_SUBJECT   -0.5
```

Taking the above example the regex is matched against the original header:

```
Subject: =?utf8?B?VGhpcyBzcGFtIGlz?=
```

- The **ADDR** modifier will perform the check on the address part of the header. Example:

```
    header      EXAMPLE_COM_SENDER   From:addr =~ /@example.com/
    describe    EXAMPLE_COM_SENDER   Message is from @example.com
    score       EXAMPLE_COM_SENDER   4
```

The specified header is parsed and the email address is extracted before the check is performed. For a header like:

```
From: Alexandru Chirila <chirila@example.com>
```

The check will be performed on *chirila@example.com*

- The **NAME** is similar to the addr modifier, but rather than checking the email address, the name of the user will be used. Example:

```
    header      EXAMPLE_COM_SENDER   From:name =~ /Alex/
    describe    EXAMPLE_COM_SENDER   Message is from Alex
    score       EXAMPLE_COM_SENDER   -4
```

Taking the above example the check is performed on the name instead of the full header (*Alexandru Chirila*)

Exists

Another modifier that can be prepended is the *exists* modifier. This will make the rule match if the message has at least one header with that name. Regardless of the header value.

Note that unlike the other modifiers this one is prepended instead of appended. Example:

```
header      DKIM_EXISTS     exists:DKIM-Signature
describe    DKIM_EXISTS     Message has DKIM signature
```

Header names

Any header name can be used when matching. However there are a few special header names that will change the behaviour.

- The **ALL** header name can be used to check all header of the message. Example:

```
    header      ONE_HEADER_WITH_SPAM   ALL =~ /spam/
    describe    ONE_HEADER_WITH_SPAM   One header had "spam"
```

- The **ToCc** header name can be used to check all the *To* and *Cc* header of the message. Example:

```
    header      ONE_EXAMPLE_RECIPIENT   ToCc =~ /@example.com/
    describe    ONE_EXAMPLE_RECIPIENT   One recipient to @example.com
```

- The **MESSAGEID** header name can be used to check various MessageID headers by a regular expression. Example:

```
    header      ONE_EXAMPLE_ID   MESSAGEID =~ /example.com/
    describe    ONE_EXAMPLE_ID   Message ID from example.com
```

MimeHeader Rule

The *mimeheader* rule is very similar to the *header* rule type, but unlike it, all the checks are done on MIME header instead of the regular message headers.

The only modifier available for the *mimeheader* is **RAW**. Examples:

```
mimeheader  HAS_PDF_ATTACHMENT  Content-Type =~ /^application\/pdf/i
describe    HAS_PDF_ATTACHMENT  Message has pdf attachments

mimeheader  HAS_PDF_ATTACHMENT  Content-Type:raw =~ /^application\/pdf/i
describe    HAS_PDF_ATTACHMENT  Message has pdf attachments
```

URI Rule

The *uri* rules type will match regular expression on all URL extracted from the message. Example:

```
uri         HAS_EXAMPLE_HTTPS   /^https:\/\/example.com$/\
describe    HAS_EXAMPLE_HTTPS   Message has HTTPS link to example.com
```

Meta Rule

The *meta* rules can be used to combine various rules in complex logic expression. This is usually used with rules that are not checked by default.

Operators that can be used in *meta* rules:

- *&&* - and operator; matches if both expression match
- || - or operator; matches if at least one expression matches
- *!* - not operator; matches if the expression doesn't match
- *()* - parentheses can be used to group multiple expressions

Examples:

```
# These rules are only checked as part of meta rules.
header      __DKIM_EXISTS           exists:DKIM-Signature
header      __EXAMPLE_COM_SENDER    From:addr =~ /@example.com/
uri         __HAS_EXAMPLE_HTTPS     /^https:\/\/example.com$/\

# The meta rules combine the above.
meta        NO_EXAMPLE_DKIM         __EXAMPLE_COM_SENDER && !__DKIM_EXISTS
describe    NO_EXAMPLE_DKIM         @example.com sender but no DKIM signature
score       NO_EXAMPLE_DKIM         5

meta        EXAMPLE_URL_SENDER      __EXAMPLE_COM_SENDER || __HAS_EXAMPLE_HTTPS
describe    EXAMPLE_URL_SENDER      example.com in sender or URL
score       EXAMPLE_URL_SENDER      2
```

```
# We can even combine meta rules in other meta rules.
meta        NO_DKIM_AND_URL         EXAMPLE_URL_SENDER && NO_EXAMPLE_DKIM
describe    NO_DKIM_AND_URL         No DKIM signature and example.com URL
score       NO_DKIM_AND_URL         3.5
```

Eval Rule

The *eval* rule type will simply call a registered evaluation function from a plugin and apply the score if function returns True. Example:

```
full        PYZOR_CHECK     eval:check_pyzor()
describe    PYZOR_CHECK     Listed in Pyzor (http://pyzor.org/)
score       PYZOR_CHECK     5.0
```

See the specific plugins documentation for all the EVAL methods it exposes and any other relevant details.

Note: When checking the method code reference ignore the **msg** and **target** parameters as those are passed by default to all eval methods.

CHAPTER 6

Plugins

To load a plugin you must add the *loadplugin* command in the configuration file. For example:

```
loadplugin pad.plugin.pyzor.PyzorPlugin
```

If the plugin is not located in the python path then you can also specify the full path to the file:

```
loadplugin MyCustomPlugin /home/pad/my_plugins/custom_plugin.py
```

Some plugins are reimplementing existing ones from SA. The full list can be seen in `pad.plugins.__init__`:

```
loadplugin Mail::SpamAssassin::Plugin::Pyzor
```

Available plugins

AutoWhiteListPlugin

Normalize scores via auto-whitelist

Example usage

```
loadplugin      pad.plugins.awl.AutoWhiteListPlugin

user_awl_sql_username username
user_awl_sql_password password
user_awl_sql_table tablename
user_awl_dsn DBI:databasetype:databasename:hostname:port

header AWL              eval:check_from_in_auto_whitelist()
describe AWL            From: address is in the auto white-list
priority AWL            1000
```

Usage

The schema for the auto whitelist:

```
CREATE TABLE `awl` (
  `username` varchar(255) NOT NULL DEFAULT '',
  `email` varchar(200) NOT NULL DEFAULT '',
```

```
  `ip` varchar(40) NOT NULL DEFAULT '',
  `count` int(11) NOT NULL DEFAULT '0',
  `totscore` float NOT NULL DEFAULT '0',
  `signedby` varchar(255) NOT NULL DEFAULT '',
  PRIMARY KEY (`username`,`email`,`signedby`,`ip`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 COMMENT='Used by SpamAssassin for the auto-whitelist functiona
```

Options

auto_whitelist_ipv4_mask_len 16 (type *int*) The AWL database keeps only the specified number of most-significant bits of an IPv4 address in its fields, so that different individual IP addresses within a subnet belonging to the same owner are managed under a single database record. As we have no information available on the allocated address ranges of senders, this CIDR mask length is only an approximation. The default is 16 bits, corresponding to a former class B. Increase the number if a finer granularity is desired, e.g. to 24 (class C) or 32. A value 0 is allowed but is not particularly useful, as it would treat the whole internet as a single organization. The number need not be a multiple of 8, any split is allowed.

auto_whitelist_factor 0.5 (type *float*) How much towards the long-term mean for the sender to regress a message. Basically, the algorithm is to track the long-term mean score of messages for the sender (C<mean>), and then once we have otherwise fully calculated the score for this message (C<score>), we calculate the final score for the message as: `C<finalscore> = C<score> + (C<mean> - C<score>) * C<factor>` So if C<factor> = 0.5, then we'll move to half way between the calculated score and the mean. If C<factor> = 0.3, then we'll move about 1/3 of the way from the score toward the mean. C<factor> = 1 means just use the long-term mean; C<factor> = 0 mean just use the calculated score.

auto_whitelist_ipv6_mask_len 48 (type *int*) The AWL database keeps only the specified number of most-significant bits

of an IPv6 address in its fields, so that different individual IP addresses within a subnet belonging to the same owner are managed under a single database record. As we have no information available on the allocated address ranges of senders, this CIDR mask length is only an approximation. The default is 48 bits, corresponding to an address range commonly allocated to individual (smaller) organizations. Increase the number for a finer granularity, e.g. to 64 or 96 or 128, or decrease for wider ranges, e.g. 32. A value 0 is allowed but is not particularly useful, as it would treat the whole internet as a single organization. The number need not be a multiple of 4, any split is allowed.

EVAL rules

Tags

AWL AWL modifier

AWLMEAN Mean score on which AWL modification is based

AWLCOUNT Number of messages on which AWL modification is based

AWLPRESCORE Score before AWL

Body Eval

Exposes several eval rules that perform checks on the body of the message.

Example usage

```
loadplugin      pad.plugins.body_eval.BodyEval

body        MPART_ALT_DIFF      eval:multipart_alternative_difference('99', '100')
describe    MPART_ALT_DIFF      HTML and text parts are different

body        MPART_ALT_DIFF_COUNT    eval:multipart_alternative_difference_count('3', '1')
describe    MPART_ALT_DIFF_COUNT    HTML and text parts are different
```

Usage

This plugin only has EVAL methods. See *Eval Rule* for general details on how to use such methods.

Options

None

EVAL rules

Tags

None

DumpText

Similar to the DumpText demo SA plugin.

Example usage

```
loadplugin      pad.plugins.dump_text.DumpText
```

Usage

Sample plugin used for testing.

Options

None

EVAL rules

Tags

None

DNSEval

Expose some eval rules that do checks on DNS lists.

Example usage

```
loadplugin      pad.plugins.dns_eval.DNSEval

header IP_IN_LIST        eval:check_rbl('example', 'example.com.', '127.0.0.10')
describe IP_IN_LIST      IP in example.com list with response 10
```

Usage

This plugin only has EVAL methods. See *Eval Rule* for general details on how to use such methods.

Options

None

EVAL rules

Tags

None

ImageInfoPlugin

Image Info plugin.

Example usage

```
loadplugin      pad.plugins.image_info.ImageInfoPlugin

body         DC_IMAGE001_GIF         eval:image_named('image001.gif')
describe     DC_IMAGE001_GIF         Contains image named image001.gif
```

Usage

This plugin exposes various methods to check image information with eval rules.

Options

None

EVAL rules

Tags

None

Header Eval

Expose some eval rules that do checks on the headers.

Example usage

```
loadplugin      pad.plugins.header_eval.HeaderEval

header DATE_IN_PAST_03_06       eval:check_for_shifted_date('-6', '-3')
describe DATE_IN_PAST_03_06     Date: is 3 to 6 hours before Received: date
```

Usage

This plugin exposes various eval rules that perform checks on the headers of the message.

See documentation for each individual rule.

Options

util_rb_tld [] (type *append_split*) Add to the TLD list

util_rb_2tld [] (type *append_split*) Add to the 2 level TLD list

util_rb_3tld [] (type *append_split*) Add to the 3 level TLD list

EVAL rules

Tags

N/A

MIME Eval

Expose some eval rules that do checks on the MIME headers

Example usage

```
loadplugin      pad.plugins.mime_eval.MIMEEval

body CHARSET_FARAWAY            eval:check_for_faraway_charset()
describe CHARSET_FARAWAY        Character set indicates a foreign language
```

Usage

This plugin exposes various eval rules that perform checks on the MIME headers of the message.

See documentation for each individual rule.

Options

None

EVAL rules

Tags

N/A

PDFInfoPlugin

This plugin helps to detect spam using attached PDF files

Example usage

```
loadplugin      pad.plugins.pdf_info.PDFInfoPlugin

body            PDF_MIME_COUNT_1        eval:pdf_count(1,3)
describe        PDF_MIME_COUNT_1        Message contains at least 1 PDF file, maximum 3.

body            PDF_IMAGE_COUNT         eval:pdf_image_count(3, 10)
describe        PDF_IMAGE_COUNT         Total number of images in PDF is between 3 and 10

body            PDF_PIX_COV             eval:pdf_pixel_coverage(100, 450)
describe        PDF_PIX_COV             Contains between 100 and 450 pixel in images

body            PDF_NAMED               eval:pdf_named('some_file.pdf')
describe        PDF_NAMED               Check if a pdf named "some_file.pdf" exists in the message.

body            PDF_NAMED_REGEX         eval:pdf_named_regex('/^(?:my|your)test\.pdf$/')
describe        PDF_NAMED_REGEX         Match if pdf is "mytest.pdf" or "yourtest.pdf"

body            PDF_MATCH_MD5           eval:pdf_match_md5('C359F8F89B290DA99DC997ED50117CDF')
describe        PDF_MATCH_MD5           Match with the PDF with that md5 hash

body            PDF_FUZZY_MD5           eval:pdf_match_fuzzy_md5('7340821445D975EEF6F5BDE2EC257900')
describe        PDF_FUZZY_MD5           Match if md5hash is in the fuzzy md5 hashes

body            PDF_MATCH_DETAIL        eval:pdf_match_details('author', '/^mobile$/')
describe        PDF_MATCH_DETAIL        Match if "mobile" is the author of the PDF file.

body            PDF_IS_ENCRYPTED        eval:pdf_is_encrypted()
describe        PDF_IS_ENCRYPTED        Match if one of the PDF files is encrypted.

body            PDF_IS_EMPTY_BODY       eval:pdf_is_empty_body(100)
describe        PDF_IS_EMPTY_BODY       Interested in PDF files larger than 100 bytes.
```

Usage

This plugin only has EVAL methods. See *Eval Rule* for general details on how to use such methods.

Options

None

EVAL rules

Tags

None

PyzorPlugin

Checks the message against the Pyzor server.

Example usage

```
loadplugin      pad.plugins.pyzor.PyzorPlugin

body        PYZOR       eval:check_pyzor()
describe    PYZOR           Listed in Pyzor (http://pyzor.org)
score       PYZOR       2
```

Usage

This plugin exposes a single eval rule that checks the message against the Pyzor server. For more information about pyzor see the Pyzor documentation

Options

use_pyzor True (type *bool*) Controls whether or not the message should be checked against the Pyzor server.

pyzor_servers ['public.pyzor.org:24441'] (type *list*) A list of Pyzor servers to check. The plugin will check ALL servers specified in this list.

pyzor_max 5 (type *int*) The minimum number of times a message needs to be reported as spam to have the rule match.

pyzor_timeout 3.5 (type *float*) The timeout for the server response.

EVAL rules

Tags

_PYZOR_DIGEST_ The pyzor digest

_PYZOR_COUNT_ The number of times the message was reported as spam on Pyzor

_PYZOR_WL_COUNT_ The number of times the message was whitelisted on Pyzor

RelayCountryPlugin

RelayCountry Plugin.

Example usage

```
loadplugin      pad.plugins.relay_country.RelayCountryPlugin
```

Usage

<Description>

Options

geodb-ipv6 GeoIPv6.dat (type *str*) <Option description>

geodb GeoIP.dat (type *str*) <Option description>

EVAL rules

None

Tags

<Describe TAGS>

ReplaceTags

This plugin allows rules to contain regular expression tags.

Example usage

```
loadplugin      pad.plugins.replace_tags.ReplaceTags

replace_start <
replace_end   >
replace_tag   A       [a@]
replace_tag   G       [gk]
replace_tag   I       [il|!1y\?\xcc\xcd\xce\xcf\xec\xed\xee\xef]
replace_tag   R       [r3]
replace_tag   V       (?:[vu]|\\\/)
replace_tag   SP      [\s~_-]

body          VIAGRA_OBFU     /(?!viagra)<V>+<SP>*<I>+<SP>*<A>+<SP>*<G>+<SP>*<R>+<SP>*<A>+/i
describe      VIAGRA_OBFU     Attempt to obfuscate "viagra"
replace_rules VIAGRA_OBFU
```

Usage

After configuring the replacement tags, the tag can then be used in any regular expression rule. By adding the extra *replace_rules NAME* line.

Options

replace_tag [] (type *append*) Assign a valid regular expression to tagname.

replace_pre [] (type *append*) Assign a valid regular expression to tagname. The expression will be placed before each tag that is replaced.

replace_post [] (type *append*) Assign a valid regular expression to tagname. The expression will be placed between each two immediately adjacent tags that are replaced.

replace_inter [] (type *append*) Assign a valid regular expression to tagname. The expression will be placed after each tag that is replaced.

replace_rules [] (type *append_split*) Specify a list of symbolic test names (separated by whitespace) of tests which should be modified using replacement tags. Only simple regular expression body, header, uri, full, rawbody tests are supported.

replace_end > (type *str*) String(s) which indicate the end of a tag inside a rule. Only tags enclosed by the start and end strings are found and replaced.

replace_start < (type *str*) String(s) which indicate the start of a tag inside a rule. Only tags enclosed by the start and end strings are found and replaced.

EVAL rules

None

Tags

None

Short Circuit

This plugin implements simple, test-based shortcircuiting. Short-circuiting a test will force all other pending rules to be skipped, if that test is hit.

Example usage

```
loadplugin      pad.plugins.short_circuit.ShortCircuit

report Details: _SCORE_/_REQD_ (_SCTYPE_)
add_header all Status "_YESNO_, score=_SCORE_ shortcircuit=_SCTYPE_"

shortcircuit_spam_score 100
shortcircuit_ham_score -100

body            TEST_RULE   /test/
describe        TEST_RULE   Test Rule
```

```
score           TEST_RULE   0.01
shortcircuit    TEST_RULE   on
```

Usage

The plugin adds a new *rule option* *shortcircuit*. To short circuit a rule simply add to the configuration file:

```
shortcircuit    <rule identifier>   [on|off|ham|spam]
```

Depending on the short-circuit type you select, the following behaviour is applied:

on If the rule matches the message the all following rules are skipped. No adjustments are done to the message score and the final result is whatever the total is at that point.

off Disables short-circuiting. The rule simply behaves as normal.

spam If the rule matches the message the all following rules are skipped. The message score is adjusted by adding the value of *shortcircuit_spam_score*.

ham If the rule matches the message the all following rules are skipped. The message score is adjusted by adding the value of *shortcircuit_ham_score*.

Options

shortcircuit_spam_score 100.0 (type *float*) The score applied for short-circuited rules with the *spam* type

shortcircuit_ham_score -100.0 (type *float*) The score applied for short-circuited rules with the *ham* type

EVAL rules

None

Tags

SCRULE The name of the rule that caused the short-circuit. This gets the value *none* if there was no such rule.

SCTYPE The type of short-circuit used. This can have the following values: on, off, ham or spam.

_SC_ Combines the other two tags for convenience. Equivalent to *_SCRULE_ (_SCTYPE_)*

TextCatPlugin

Detect the language of the message.

Current available languages:

af ar bg bn ca cs cy da de el en es et fa fi fr gu he hi hr hu id it ja kn ko lt lv mk ml mr ne nl no pa pl pt ro ru sk sl so sq sv sw ta te th tl tr uk ur vi zh-cn zh-tw

Example usage

```
loadplugin      pad.plugins.textcat.TextCatPlugin
```

Usage

N/A

Options

textcat_optimal_ngrams 0 (type *int*) <Option description>

textcat_max_ngrams 400 (type *int*) <Option description>

ok_languages all (type *list*) <Option description>

textcat_acceptable_prob 0.7 (type *float*) <Option description>

inactive_languages (type *list*) <Option description>

textcat_acceptable_score 1.05 (type *float*) <Option description>

textcat_max_languages 5 (type *int*) <Option description>

EVAL rules

Tags

N/A

URIDetailPlugin

URIDetail Plugin.

Example usage

```
loadplugin      pad.plugins.uri_detail.URIDetailPlugin
```

Usage

N/A

Options

uri_detail [] (type *list*) <Option description>

EVAL rules

None

Tags

None

WhiteListSubjectPlugin

Whitelist Subject plugin.

Example usage

```
loadplugin      pad.plugins.whitelist_subject.WhiteListSubjectPlugin
```

Usage

N/A

Options

blacklist_subject [] (type *list*) <Option description>

whitelist_subject [] (type *list*) <Option description>

EVAL rules

Tags

N/A

SPF Plugin

This plugin checks a message against Sender Policy Framework (SPF) records published by the domain owners in DNS to fight email address forgery and make it easier to identify spams.

Example usage

Evaluation of an SPF record can return any of these results: **Pass**

The SPF record designates the host to be allowed to send. Action: accept.

Fail The SPF record has designated the host as NOT being allowed to send. Action: reject.

SoftFail The SPF record has designated the host as NOT being allowed to send but is in transition. Action: accept but mark.

Neutral The SPF record specifies explicitly that nothing can be said about validity. Action: accept.

None The domain does not have an SPF record or the SPF record does not evaluate to a result. Action: accept.

PermError A permanent error has occured (eg. badly formatted SPF record). Action: unspecified.

TempError A transient error has occured. Action: accept or reject

```
loadplugin      pad.plugins.spf.SpfPlugin

header SPF_PASS             eval:check_for_spf_pass()
header SPF_NEUTRAL          eval:check_for_spf_neutral()
header SPF_FAIL             eval:check_for_spf_fail()
header SPF_SOFTFAIL         eval:check_for_spf_softfail()
```

Usage

This plugin has EVAL methods. See *Eval Rule* for general details on how to use such methods.

Options

whitelist_from_spf address@example.com <Not available yet>

spf_timeout n (default: 5) How many seconds to wait for an SPF query to complete, before scanning continues without the SPF result.

ignore_received_spf_header (False|True) (default: False) By default, to avoid unnecessary DNS lookups, the plugin will try to use the SPF results found in any *Received-SPF* headers it finds in the message that could only have been added by an internal relay

Set this option to True to ignore any *Received-SPF* headers present and to have the plugin perform the SPF check itself.

use_newest_received_spf_header (False|True) (default: False) By default, when using *Received-SPF* headers, the plugin will attempt to use the oldest (bottom most) *Received-SPF* headers, that were added by internal relays, that it can parse the results from since they are the most likely to be accurate.This is done so that if you have an incoming mail setup where one of your primary MXes doesn't know about a secondary MX (or your MXes don't know about some sort of forwarding relay that SA considers trusted+internal) but SA is aware of the actual domain boundary (internal_networks setting) SA will use the results that are most accurate.

Use this option to start with the newest (top most) *Received-SPF* headers, working downwards until results are successfully parsed.

EVAL rules

Tags

None

WLBLEvalPlugin

This plugin checks if from addresses and to addresses are in options list by domanin, IP and URI.

Example usage

```
loadplugin      pad.plugins.wlbl_eval.WLBLEvalPlugin
```

Usage

N/A

Options

blacklist_from [] (type *append_split*) Used to specify addresses which send mail that is often tagged (incorrectly) as non-spam, but which the user doesn't want. Same format as whitelist_from.

whitelist_from [] (type *append_split*) Used to whitelist sender addresses which send mail that is often tagged (incorrectly) as spam.

whitelist_to [] (type *append_split*) If the given address appears as a recipient in the message headers (Resent-To, To, Cc, obvious envelope recipient, etc.) the mail will be whitelisted. There are three levels of To-whitelisting, whitelist_to, more_spam_to and all_spam_to. Users in the first level may still get some spammish mails blocked, but users in all_spam_to should never get mail blocked.

all_spam_to [] (type *append_split*) See above.

more_spam_to [] (type *append_split*) See above.

blacklist_to [] (type *append_split*) If the given address appears as a recipient in the message headers (Resent-To, To, Cc, obvious envelope recipient, etc.) the mail will be blacklisted.

def_whitelist_from_rcvd [] (type *list*) Same as whitelist_from_rcvd, but used for the default whitelist entries in the OrangeAssassin distribution. The whitelist score is lower, because these are often targets for spammer spoofing.

whitelist_from_rcvd [] (type *list*) Works similarly to whitelist_from, except that in addition to matching a sender address, a relay's rDNS name or its IP address must match too for the whitelisting rule to fire. The first parameter is a sender's e-mail address to whitelist, and the second is a string to match the relay's rDNS, or its IP address.

whitelist_allow_relays [] (type *append_split*) Specify addresses which are in whitelist_from_rcvd that sometimes send through a mail relay other than the listed ones.

enlist_uri_host [] (type *list*) Adds one or more host names or domain names to a named list of URI domains.

delist_uri_host [] (type *list*) Removes one or more specified host names from a named list of URI domains.

blacklist_uri_host [] (type *list*) Is a shorthand for a directive: enlist_uri_host (BLACK) host.

whitelist_uri_host [] (type *list*) Is a shorthand for a directive: enlist_uri_host (WHITE) host

util_rb_tld [] (type *append_split*) This option maintains list of valid TLDs in the RegistryBoundaries code.

util_rb_2tld [] (type *append_split*) This option maintains list of valid 2nd-level TLDs in the RegistryBoundaries code.

util_rb_3tld [] (type *append_split*) This option maintains list of valid 3rd-level TLDs in the RegistryBoundaries code.

EVAL rules

Tags

Non

Razor2Plugin

Checks the message against the Razor server.

Example usage

```
loadplugin      pad.plugins.pyzor.Razor2Plugin

body        RAZOR2      eval:check_razor2()
describe    RAZOR2          Listed in Razor2 (http://razor.sf.net/)
score       RAZOR2       1
```

Usage

This plugin exposes a two eval rules that checks the message against the Razor server. "check_razor2_range" method it is implemented, but in order to verify a message, you can use PyzorPlugin. For more information about pyzor see the Razor documentation

Options

use_razor2 True (type *bool*) Controls whether or not the message should be checked against the Razor server.

razor_config "" (type *str*) Define the filename used to store Razor's configuration settings. Currently this is left to Razor to decide.

razor_timeout 5 (type *int*) How many seconds you wait for Razor to complete before you go on without the results.

EVAL rules

Tags

None

SpamCopPlugin

SpamCop is a service for reporting spam. SpamCop determines the origin of unwanted email and reports it to the relevant Internet service providers. Note that spam reports sent by this plugin to SpamCop each include the entire spam message.

Example usage

```
loadplugin      pad.plugins.spam_cop.SpamCopPlugin
```

Usage

N/A

Options

spamcop_from_address "" (type *str*) This address is used during manual reports to SpamCop as the From: address. You can use your normal email address. If this is not set, a guess will be used as the From: address in SpamCop reports.

spamcop_to_address "spamassassin-submit@spam.spamcop.net" (type *str*) Your customized SpamCop report submission address. You need to obtain this address by registering at http://www.spamcop.net/. If this is not set, SpamCop reports will go to a generic reporting address for OrangeAssassin users and your reports will probably have less weight in the SpamCop system.

spamcop_max_report_size 50 (type *int*) Messages larger than this size (in kilobytes) will be truncated in report messages sent to SpamCop. The default setting is the maximum size that SpamCop will accept at the time of release.

dont_report_to_spamcop False (type *bool*)

EVAL rules

None

Tags

None

FreeMailPlugin

Checks the headers for indication that sender's domain is that of a site offering free email services.

Example usage

```
loadplugin      pad.plugins.free_mail.FreeMailPlugin

header CHECK_FREEMAIL_FROM                eval:check_freemail_from()
header CHECK_FREEMAIL_FROM_REGEX          eval:check_freemail_from('\d@')

header CHECK_FREEMAIL_BODY                eval:check_freemail_body()
header CHECK_FREEMAIL_BODY_REGEX          eval:check_freemail_body('\d@')

header CHECK_FREEMAIL_HEADER              eval:check_freemail_header('From')
header CHECK_FREEMAIL_HEADER_REGEX        eval:check_freemail_header('From', '\d@')

header CHECK_FREEMAIL_REPLY_TO eval:check_freemail_replyto('replyto')
header CHECK_FREEMAIL_REPLY eval:check_freemail_replyto('reply')

util_rb_tld com
util_rb_tld net

freemail_domains example.com
freemail_add_describe_email 1

report _REPORT_
report _SCORE_
report _TESTS_


The output:

* 1.0 CHECK_FREEMAIL_BODY Body has freemails
    (test[at]example.com)
* 1.0 CHECK_FREEMAIL_REPLY Different freemails in reply header and body
    (sender[at]example.com test[at]example.com)
* 1.0 CHECK_FREEMAIL_FROM Sender address is freemail
    (sender[at]example.com)
* 1.0 CHECK_FREEMAIL_HEADER Header From is freemail
    (sender[at]example.com)
4.0
CHECK_FREEMAIL_BODY,CHECK_FREEMAIL_REPLY,CHECK_FREEMAIL_FROM,CHECK_FREEMAIL_HEADER
```

Usage

If From-address is freemail, and Reply-To or address found in mail body is

a different freemail address, return success.

Options

freemail_domains [] (type *append_split*) List of domains to be used in checks. Regexp is not supported, but following wildcards work:

? for single character (does not match a dot) * for multiple characters (does not match a dot)

For example: freemail_domains hotmail.com hotmail.co.?? yahoo.* yahoo.*.*

freemail_whitelist [] (type *append_split*) Emails or domains listed here are ignored (pretend they are not freemails). No wildcards!

freemail_max_body_emails 5 (type *int*)

freemail_max_body_freemails 3 (type *int*)

freemail_skip_when_over_max True (type *bool*)

freemail_skip_bulk_envfrom True (type *bool*)

freemail_add_describe_email True (type *bool*) When this option is True (enabled), the report also contains the email that matched.

For example:

freemail_add_describe_email 1

- **1.0 CHECK_FREEMAIL_FROM Sender address is freemail**

(sender[at]example.com)

AND

freemail_add_describe_email 0

- 1.0 CHECK_FREEMAIL_FROM Sender address is freemail

util_rb_tld [] (type *append_split*) List of valid tlds (level 1)

For example: .com, .ro

util_rb_2tld [] (type *append_split*) List of valid tlds (level 2)

For example: .co.uk, .org.uk

util_rb_3tld [] (type *append_split*) List of valid tlds (level 3)

For example: .sa.edu.au

EVAL rules

Tags

None

DKIMPlugin

This plugin performs verifications on DKIM signature

Example usage

```
loadplugin      pad.plugins.dkim.DKIMPlugin

adsp_override example.com custom_high
whitelist_from_dkim user@example.com
def_whitelist_from_dkim  *@example.com example.com


full   DKIM_SIGNED           eval:check_dkim_signed("example.com")
full   DKIM_VALID            eval:check_dkim_valid("example.com")
full   DKIM_VALID_AU         eval:check_dkim_valid_author_sig('example.com')

header DKIM_ADSP_NXDOMAIN    eval:check_dkim_adsp('*', 'example.com')
header DKIM_ADSP_ALL         eval:check_dkim_adsp('A')
header DKIM_ADSP_DISCARD     eval:check_dkim_adsp('D')
header DKIM_ADSP_CUSTOM_LOW  eval:check_dkim_adsp('1')
header DKIM_ADSP_CUSTOM_MED  eval:check_dkim_adsp('2')
header DKIM_ADSP_CUSTOM_HIGH eval:check_dkim_adsp('3')

header USER_IN_DKIM_WL      eval:check_for_dkim_whitelist_from()
header USER_IN_DEF_DKIM_WL  eval:check_for_def_dkim_whitelist_from()
```

Usage

<Description>

Options

whitelist_from_dkim [] (type *list*) Used to whitelist sender addresses which send mail that is often tagged (incorrectly) as spam.

def_whitelist_from_dkim [] (type *append_split*) Same as 'whitelist_from_dkim', but used for the deafult whitelist entries.

unwhitelist_from_dkim [] (type *list*) Removes an email address with its corresponding signing-domain field from def_whitelist_from_dkim and whitelist_from_dkim tables, if it exists.

adsp_override [] (type *list*) To override domain's signing practices in a OrangeAssassin configuration file, specify an adsp_override directive for each sending domain to be overridden. An optional second parameter is one of the following keywords: nxdomain, unknown, all, discardable, custom_low, custom_med, custom_high. Absence of this second parameter implies discardable.

dkim_minimum_key_bits 1024 (type *int*) The smallest size of a signing key (in bits) for a valid signature to be considered for whitelisting.

EVAL rules

Tags

None

URIEvalPlugin

URIEval Plugin.

Example usage

```
loadplugin      pad.plugins.uri_eval.URIEvalPlugin
```

Usage

N/A

Options

None

EVAL rules

Tags

N/A

RelayEvalPlugin

Check the data parsed from ReceivedParser against different rules.

Evaluate a set of rules against "Received" headers, they are form a list of all the servers/computers through which the message traveled in order to reach the destination.

Example usage

```
loadplugin pad.plugins.relay_eval.RelayEval

header RCVD_HELO_IP_MISMATCH        eval:helo_ip_mismatch()
describe RCVD_HELO_IP_MISMATCH      Received: HELO and IP do not match, but should

header RCVD_NUMERIC_HELO    eval:check_for_numeric_helo()
describe RCVD_NUMERIC_HELO  Received: contains an IP address used for HELO

header __FORGED_RCVD_TRAIL  eval:check_for_forged_received_trail()

header NO_RDNS_DOTCOM_HELO  eval:check_for_no_rdns_dotcom_helo()
describe NO_RDNS_DOTCOM_HELO        Host HELO'd as a big ISP, but had no rDNS
```

Usage

This plugin only has EVAL methods. See *Eval Rule* for general details on how to use such methods.

Options

None

EVAL rules

Tags

None

AutoLearnThreshold

Implements the functionality to submit messages for learning when they fall outside the defined threshold

Example usage

```
loadplugin      pad.plugins.auto_learn_threshold.AutoLearnThreshold
bayes_auto_learn_threshold_nonspam 0.5 # optional, default is 0.1
bayes_auto_learn_threshold_nonspam 12.0 # optional, default is 12.0
bayes_auto_learn_on_error 1 # optional, default is 1
```

Usage

When this plugin is loaded after the message has been evaluated by all other plugins it will be evaluated for autolearning. It will be evalute accoring to the following rules:

It calculates the total score for the message from tests that don't have the noautolearn, userconf tflags

General requirements

- The autolearn score includes at least 3 body and 3 header tests scores (unless any test has the tflag **autolearn_force** in which case the header

  and body tests requirement drops to -99)
- The bayes plugin classified the message differently than this plugin (unless **bayes_auto_learn_on_error** option is set to 0)

Case 1

- The message score was higher than the required score
- The message is considered spam by the autolearn plugin (the autolearn score is higher than the spam threshold)
- The score from tests with the **learn** tflag is at least -1

Case 2

- The message score was lower than the required score
- The message is considered ham by the autolearn plugin (the autolearn score is lower than the ham threshold)
- The score from tests with the **learn** tflag is at least 1

Options

bayes_auto_learn_threshold_nonspam 0.1 (type *float*) Messages that score below this value will be submitted for learning as HAM

bayes_auto_learn_threshold_spam 12.0 (type *float*) Messages that score over this value will be submitted for learning as SPAM

bayes_auto_learn_on_error 0 (type *bool*) Messages will be submitted for learning only if Bayes disagrees with the classification

EVAL rules

This plugin doesn't expose any eval rules

Tags

None

Plugin reference

plugins Package

Module `base`

Plugin `awl`

Plugin `body_eval`

Plugin `dump_text`

Plugin `dns_eval`

Plugin `image_info`

Plugin `header_eval`

Plugin `mime_eval`

Plugin `pdf_info`

Plugin `pyzor`

Plugin `relay_country`

Plugin `replace_tags`

Plugin `short_circuit`

Plugin `textcat`

Plugin `uri_detail`

Plugin `whitelist_subject`

Plugin `spf`

Plugin `wlbl_eval`

Plugin `razor2`

Plugin `free_mail`

Plugin `spam_cop`

Plugin `dkim`

Plugin `uri_eval`

Plugin `relay-eval`

members

undoc-members

show-inheritance

CHAPTER 7

Reference

pad Package

protocol Package

protocol Package

base Module

check Module

noop Module

process Module

tell Module

rules Package

rules Package

base Module

body Module

eval_ Module

full Module

header Module

meta Module

parser Module

ruleset Module

uri Module

conf Module

config Module

context Module

errors Module

message Module

regex Module

server Module

scripts Package

match Module

oad Module

Starts the OrangeAssassin daemon.

oad.**main**()

oad.**run_daemon**(*args*)
Start the daemon.

compile Module

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

O

M

O

R